Travaux Dirigés d'algorithmique n°6

Cours d'Informatique de Deuxième Année

—DEUG Sciences S3-MI—

Listes chaînées (suite)

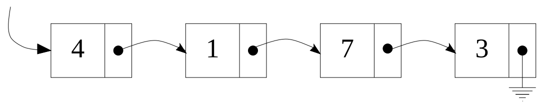
Nous implémenterons dans ce TD quelques opérations supplémentaires sur les listes chaînées.

Nous utiliserons les définitions suivantes pour mettre en œuvre la notion de "liste chaînée d'entiers".

```
typedef struct _cellule
        { int Val;                     /* donnee stockee : un entier.       */
          struct _cellule * PtrSuiv;  /* pointeur sur la cellule suivante. */
        } Cellule;                     /* definition d'un nouveau type.     */

typedef Cellule * Liste;              /* definition d'un nouveau type.     */
```

Ainsi, la liste d'entiers $(4, 1, 7, 3)$ est représentée par la liste chaînée d'entiers suivante :



▸ **Exercice 1.** ***(Duplication d'une liste)*** *Écrire une fonction* `DupListe` *qui crée une liste identique à celle fournie en argument de la fonction.*

▸ **Exercice 2.** ***(Suppression dans une liste)***

- *Écrire une fonction* `EffaceUn` *qui supprime la première occurrence d'un entier* `n` *dans une liste chaînée.*
- *Écrire une fonction* `EffaceTout` *qui supprime toutes les occurrences d'un entier* `n` *dans une liste chaînée.*

▸ **Exercice 3.** ***(Libération d'une liste)*** *Écrire une fonction* `LibereListe` *qui libère tout l'espace mémoire occupé par une liste chaînée.*

▸ **Exercice 4.** ***(Inversion d'une liste)*** *Écrire une fonction* `Miroir` *qui calcule l'image miroir de la liste chaînée passée en paramètre. Cette fonction ne doit pas détruire la liste chaînée d'origine mais créer une nouvelle liste totalement indépendante.*

▸ **Exercice 5.** ***(Inversion d'une liste sur place)*** *Écrire une fonction* `MiroirSurPlace` *qui calcule l'image miroir de la liste chaînée passée en paramètre. Cette fonction ne doit pas créer de nouvelles cellules. Elle modifiera simplement le chaînage de la liste chaînée d'origine.*